Lecture 7: Cuckoo Hashing

Prof. Eric Price *Scribe: Marlan McInnes-Taylor*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

We have thus far examined two types of hashing: Standard and Two-Choice. Two-Choice Hashing offers a better upper bound compared to Standard Hashing in the high probability lookup case. Today, we will examine Cuckoo Hashing [PR01], which offers an even better bound in the high probability lookup case.

1.1 Bounds

	Standard	**Two-Choice**	**Cuckoo**
Keys	n	n	n
Space	$O(n)$	$O(n)$	$O(n)$
Expected lookup	$O(1)$	$O(1)$	$O(1)$
Worst case lookup	$O\left(\frac{\log(n)}{\text{loglog}(n)}\right)$	$O(\text{loglog}(n))$	$O(1)$
Expected insertion	$O(1)$	$O(1)$	$O(1)$
Worst case insertion	$O(1)$	$O(1)$	$O(\log(n))$

2 Problem Setting

In Two-Choice hashing we utilized the balls and bins analogy. We shall reframe our conceptualization in terms of a graph. Think of T-C-H ball insertion as edge insertion into a random graph.

(TCH) hash m balls into n bins $\rightarrow$ **(Cuckoo)** insert m random edges in a directed n-vertex graph

3 Algorithm

Cuckoo hashing utilizes two hash functions, as well as an eviction mechanism in its scheme. Each location in the hash table can contain at most one item.

Let:

T be a table
x be an item to insert
$h_1(\cdot),\ h_2(\cdot)$ be hash functions s.t. $\forall x\ h_1(x) \neq h_2(x)$

3.1 Insertion

For each x:

1. Compute $h_1(x),\ h_2(x)$
2. Check if $T[h_1(x)], T[h_2(x)]$ are occupied.
3. Insert x into the table such that:
    - If both indices are unoccupied, randomly select one of the indices for insertion
    - If only one index is unoccupied, insert at that index
    - If both locations already contain an element, then randomly evict one element, call it x', from either location and insert x in its place.
4. If the insertion caused an eviction, reinsert x'
5. Continue until all elements are placed successfully

Occasionally, the algorithm cannot place every element into the table; in other words it is stuck in an insertion/eviction loop. Graphically, this occurs when the vertices and edges form a barbell. This triggers a table rebuild, where new hash functions are selected, and all elements are rehashed.

4 Analysis

Let:

$G = (V, E)$ be a graph representing the Cuckoo hash table
m be the number of edges/items
n be the number of vertices/indices

The insertion procedure will be analyzed in a graphical context by first examining how often we must rebuild our hash table, followed by how costly a rebuild is in terms of time.

Need to show:

- Pr[a cycle in G], which will bound the barbell and thus rebuild probability
- Good time to build table

4.1 Lookup

For any given lookup, we simply check two indices(vertices), so the worst case lookup cost is *O(1)*.

4.2 Rebuild Occurrence

A rebuild is triggered whenever an item cannot be inserted into the table. As mentioned in Section 3, this occurs when the graph contains a barbell, which in this application is two cycles connected by a single edge. Therefore, we will use cycle existence to bound our rebuild occurrence.

4.2.1 Length k cycle existence

$$\begin{aligned}
\Pr[\text{a length } k \text{ cycle exists}] &= (\#\text{ length } k \text{ cycles}) \cdot \Pr[\text{particular length } k \text{ cycle exists}] \\
&= \left[\frac{\binom{n}{k} \cdot k!}{2k}\right] \cdot \left(\frac{m}{\binom{n}{2}}\right)^k \\
&\leq \left[\binom{n}{k}(k-1)!\right] \cdot \left(\frac{m}{\binom{n}{2}}\right)^k \\
&\leq n^k \cdot \left(\frac{m}{\binom{n}{2}}\right)^k \\
&= \left(\frac{2m}{n-1}\right)^k
\end{aligned}$$

While this suffices to bound the probability of encountering a particular length k cycle, our barbell bound must account for cycles of *any* length.

Note: $\Pr[\text{given edge exists}] = 1 - \left(1 - \frac{1}{\binom{n}{2}}\right)^m \leq \frac{m}{\binom{n}{2}}$

4.2.2 Barbell existence

$$\begin{aligned}
\Pr[\text{a barbell exists}] &\leq \sum_{k=2}^{n} \Pr[\text{length } k \text{ cycle exists}] \\
&\leq \sum_{k=2}^{n} \left(\frac{2m}{n-1}\right)^k \\
&\leq \left(\frac{2m}{n-1}\right)^2
\end{aligned}$$

The final inequality is due to the $k = 2$ term dominating the summation. Therefore if we have $n = 15m$ vertices in our graph, the probability of rebuilding is at most $\frac{1}{49}$.

4.3 Build Time

Intuitively, the time to build the table should be bounded by the time spent inserting each item.

$$\begin{aligned}\mathbb{E}[\text{time to build table}] &\leq \sum_{i=1}^{m} \mathbb{E}[\text{time to place item } i] \\ &\leq \sum_{i=1}^{m} \mathbb{E}[\text{size of component touched by item } i]\end{aligned}$$

Claim: For any fixed vertex v, $\mathbb{E}[\text{size of component containing } v] = O(1)$.

To further the analysis, consider the Erdős-Renyi model $G(n, p)$, where n denotes the number of vertices in the graph, and p denotes the (independent) probability an edge is included in the graph. Using this framework, we can characterize the size of a connected component in the graph using the Galton-Watson branching process [WG75]. For any given vertex v there are at most $n-1$ possible neighbors, each with probability p. For any vertex u connected to v, it can also have $n-1$ possible neighbors, and so forth. This structure can be thought of as an infinite tree.

Let:

$$f(n, p) \text{ be the expected component size of a given vertex}$$

Where:

$$\begin{aligned}f(n,p) &\leq 1 + (n-1)p \cdot f(n-1,p) \\ &= 1 + p(n-1) + p^2(n-1)(n-2) + p^3(n-1)(n-2)(n-3) + ... \\ &\leq 1 + np + (np)^2 + (np)^3 + ... \\ &\leq \frac{1}{1-np}\end{aligned}$$

Recall: $\Pr[\text{given edge exists}] = 1 - \left(1 - \frac{1}{\binom{n}{2}}\right)^m \leq \frac{m}{\binom{n}{2}}$

Utilizing the above probability, we have:

$$f(n,p) \backsim \frac{1}{1 - \frac{2m}{n}} = O(1)$$

Therefore, for a table being built with m items, each with an expected insertion time of *O(1)*, we have a build time of *O(m)*.

References

[PR01] Rasmus Pagh, Flemming Friche Rodler. Cuckoo Hashing. *Algorithms — ESA 2001.*, Lecture Notes in Computer Science, vol 2161, Springer, Berlin, Heidelberg, 2001. https://doi.org/10.1007/3-540-44676-1_10

[ER60] Paul Erdős, Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[WG75] Henry William Watson, Francis Galton. On the probability of the extinction of families. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 4:138-144, 1875.